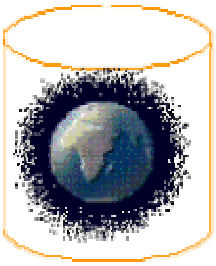

DBGlobe

IST-2001-32645

DBGlobe: A Data Centric Approach to Global Computing

Deliverable D15

Prototype System for Location-Aware Query Processing (description and manual)

Report Version: 1.0
Report Preparation Date: December 15, 2003
Delivery Date to EC: December 17, 2003
Classification: Public
Contract Start Date: 1/1/2002 **Duration**: 36 months
Estimated Person Months: 10 **Total Number of Pages:** 34
Project Coordinator: University of Ioannina
Contributing Partners: University of Ioannina, Institute National de Recherché en Informatique et en Automatique, Computer Technology Institute, Research Center of the Athens University of Economics and Business, University of Cyprus, University of California, Riverside

Project funded by the European Community under the "Information Society Technologies" Programme (1998-2002)

Abstract

Mobile devices (or PMOs) and wireless networking brings about a new computing paradigm where location and wireless services are of central importance. Mobile devices as they relocate themselves create ad hoc databases around their current location. A client usually requests a particular service that is of close proximity, usually within his local Cell (or CAS). We can thus view location as a concept around which we pose queries. Concept is a semantic notion and describes a specific property, for example, "travelling", "weather", "taxi reservation" etc. In this deliverable, we provide a description of the proof of concept. In particular, we will present a prototype implementation of an application that uses the "communities" that were described in deliverable D14.

Executive Summary

This report is a product of WP5 and aims at providing a proof of concept prototype for the Location Aware Ad-hoc Databases and the Query Processing mechanisms of the DBGlobe framework. The DBGlobe framework aims to provide an efficient way of service and data sharing among a variety of mobile devices. The goal of this report is to present a proof of concept for the mechanism of ad hoc database creation (i.e. communities) and the support of context aware queries as described in the previous deliverable (deliverable D14).

Contents

Abstract **2**

Executive Summary **2**

1. Introduction **4**

2. Using the Prototype Application with Communities **6**

3. Implementation Issues **9**

4. User's Manual **11**

4.1 DBGlobe System Manual *11*

Creating a new account *12*

Login process *13*

Using the DBGlobe system *14*

View Communities *15*

Register a service *16*

Find a service *17*

View service search results *18*

Logout *19*

4.2 Manual for the Proof-of-Concept Prototype **20**

Register to the system *21*

Login to the application *22*

View your group and friends *23*

View friends/partner profile *24*

Application Options *25*

Friend Management *26*

Service Management *28*

Search for services *29*

Register a new service *31*

Logout *32*

Conclusions **33**

1. Introduction

A concept central to the DBGlobe system is the concept of a context-aware ad-hoc database. This concept and the associated query processing mechanisms were presented in Deliverable D14. The objective of the query mechanism is to support context continuous and containment queries as efficient as possible around the notion of community taking into account problems such as disconnections and device limitations. In order to test and provide a proof of concept prototype, we have developed and experimented with a prototype implementation that uses this mechanism. To this end, we created "SpotMe", an application that utilizes the idea of communities as this was presented in D14. The general operation and functionality of "SpotMe" is to enable a user to find out where his friends/colleagues/partners are, what services are offered to him (either by his friends/partners or others that registered with the DBGlobe system) and allow him to advertise/post his own services. In detail, "SpotMe" aims at providing its users with the following capabilities:

- Location Management (for example, find the nearest friend/service)
- Notion of community (find services of a given type)
- Track local (to the user) services
- Find any service (based on keywords)
- Allow users to register their services.

SpotMe is a context aware application built around web services. Context aware application servers are connected with the DBGlobe system by calling the appropriate web-services (e.g. CAS web-service, or Community web-service). In this way, any SpotMe client that enters a DBGlobe covered area can use the SpotMe application (as shown in Figure 1.1).

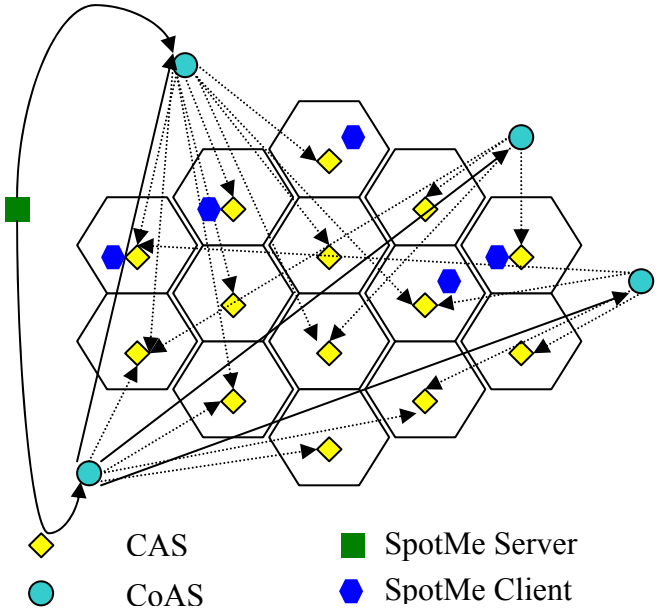

Figure 1.1: Geographical distribution of SpotMe Clients & Servers

In the remainder of this report, we provide a detailed description of the communication model between the DBGlobe system and the applications which can be built on top of it. We focus on what features are provided by the DBGlobe system, how "SpotMe" utilizes communities and how communities enable "SpotMe" to achieve its goals.

2. Using the Prototype Application with Communities

The DBGlobe system can be seen as a platform that enables context aware applications (i.e. SpotMe) to be build over it by providing them with a set of the necessary services (i.e. locating user and services). Figure 2.1 shows the layering of the system as a whole. The context aware applications (i.e. SpotMe) can directly use (through the web services protocol) any CoAS or CAS.

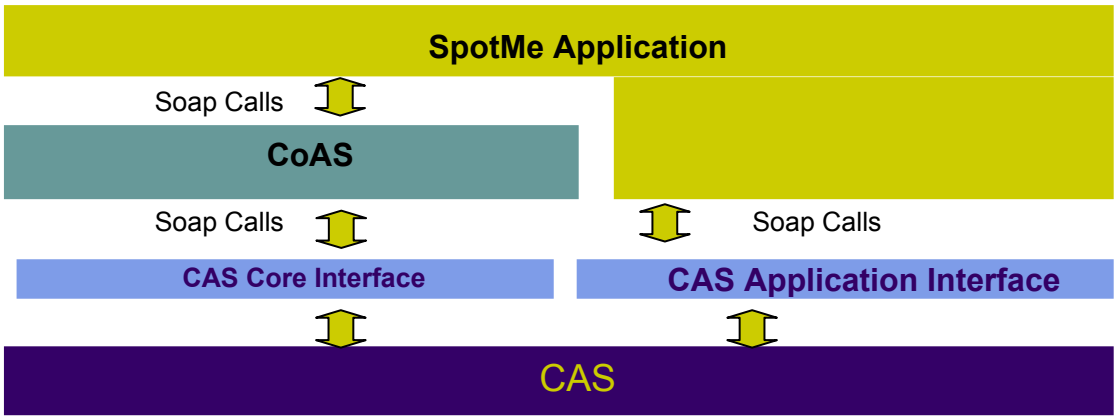

Figure 2.1: SpotMe connection with DBGlobe

In more detail, the "SpotMe" application is comprised of an application server and its clients. The application server (SpotMe server) is connected with the DBGlobe system (Figure 2.1). As depicted in Figure 2.1, our prototype implementation is not part of the DBGlobe system; instead it is an application that utilizes DBGlobe.

"SpotMe" helps users locate their online friends as well as nearby services. To achieve this, it utilizes the DBGlobe mechanisms for location management and service discovery. More specifically, since DBGlobe uses the communities' mechanism to answer questions that refer to multiple cells (as it was described in D14), the same applies to the SpotMe application. Additionally, the SpotMe application (the server of the application) can request from the DBGlobe system to add user services or ask to find a service that will be answered by the related community.

The various steps needed to be taken and functions that are offered to make the SpotMe application operational are the following:

1. **Initialization of a SpotMe server**. During the initialization, the server needs to know at least one CAS in order to be able to connect with the DBGlobe system.
2. **Initialization of SpotMe clients.** The client just needs to know the address of a SpotMe server in order to login and start using the application. Please note that in order to connect with the network, a SpotMe client must be within range of a DBGlobe cell and therefore a CAS.
3. **Login of a SpotMe client.** During its login, a SpotMe client connects to a SpotMe server.
4. **The functions a SpotMe client can perform** (through the SpotMe application) are the following:
    a. Access to local services (*exhibits location management*). In order to do so, a client needs to know which services are local to him. Thus, upon such a request, the SpotMe server utilizes the location management of DBGlobe to both discover the requester location and the local to the requestor services (*"Select services that contain 'keywordslist' near by 'Cyprus'"*). DBGlobe in turn will provide the requested information by contacting the user's home location, get the user's current location and find all services registered to the user's current location.
    b. View cooperating services (*exhibits location management and communities*). Cooperating services are the services offered by a friend (or another SpotMe client that is in the same group with the requesting client). An example of this would be for a SpotMe client that belongs to the programmer's community to ask its fellow programmers for a piece of Java code (*"Select service where contains 'java, code'"*). The answer to such a question can be easily given, if we have in place a community for programmers and their services. The only necessary action is to forward this query to the related CoAS(s). So, a SpotMe client will ask (through the application and thus the SpotMe server) the DBGlobe system to find the services of the programmer's group. The DBGlobe system, in turn, will forward this query to the relative community and get the requested answer.

c. Register a service to the system (*exhibits communities*). A SpotMe client may offer some services to others. To make this possible the client registers the SpotMe server of its services. The server then, registers those services to the DBGlobe system in order to be used by other users.

5. **The functions of a SpotMe server are the following**:
   a. Create group of friends/cooperators. Furthermore SpotMe servers are also able to:
      i. *Add/Delete/Edit a friend/cooperator*. This is achieved by asking the SpotMe server to add/delete/edit him.
      ii. *Show the user location.* This capability exhibits location management. DBGlobe (with the aid of communities) answers these questions.
   b. *Find a service based on keywords* (*exhibits communities*). In order to find a service the SpotMe servers ask directly the relative communities or through the DBGlobe system for a list of services that match the given keywords.
6. **Get Services that belong to a specific category (*exhibits communities*)**. Again to achieve this, DBGlobe communities are utilized. When a SpotMe client asks for services it does so through the SpotMe application. What that means is that it requests from the SpotMe server to find him all the services belonging to the specific category. SpotMe servers forward the request to the appropriate community or to the DBGlobe system and get the wanted answer which is then returned to the client.

All the previous capabilities are very important and necessary for any context aware application, their availability through the DBGlobe system enables the applications to be built rapidly and without the need to implement the whole system from scratch. Additionally, due to the use of web service protocols, the application can be implemented in a variety of programming languages without compatibility problems. In the following section, we discuss in more detail such implementation issues.

3. Implementation Issues

The DBGlobe architecture can be seen as a middleware that enables the fast and efficient implementation of context aware applications. In this section, we present implementation issues and details for creating a context aware application over the DBGlobe system.

The DBGlobe system provides a web service interface that enables any application, written in any programming language, to have access to the services it provides by using SOAP [5] messages. Example services provided by CAS(s) include the following:

- Find the user location
- Update the user location
- Find local to the user services
- Find any service that matches with a list of keywords
- Register a service

Moreover, Communities have also a web service interface that enables applications to access them by using SOAP messages. In our case, we use axis technology [11] as a web service engine.

SOAP [5] is a lightweight protocol for exchange of information in a decentralized, distributed environment. It is an XML based protocol that consists of three parts: an envelope that defines a framework for describing what is in a message and how to process it, a set of encoding rules for expressing instances of application-defined datatypes, and a convention for representing remote procedure calls and responses. In order to handle the SOAP messages, we use Axis. Axis is essentially a SOAP engine – a framework for constructing SOAP processors such as clients, servers, gateways, etc. The current version of Axis is based on Java.

Because DBGlobe uses AXML framework – that allows using arbitrary SOAP based web services and also pecifying and using more elaborate services that may query and update AXML documents [8,9,10] – we can also use this framework to establish service calls to the registered services.

The AXML framework is centered on AXML documents, which are XML documents that contain service calls to web services. The AXML framework is ideal

for data and web service integration. The context of an AXML document is dynamic, since you can specify when a service call should be activated and for how long the results are valid. AXML allows also calling continuous services that provide stream of answers and service supporting intentional data (AXML documents including service calls) as parameters and/or result.

Our application prototype has been implemented by using java technology (JSDK 1.4.1), web server-Tomcat 5 and "Mysql" as database technology. The Tomcat service is responsible to execute the dynamic pages and the Mysql database management system is used for storing information about the users, their friends and their options. The rest of the information (e.g. user services, user location) is provided by the DBGlobe system; the application just calls the appropriate service. Communities have also been implemented using java (JSDK 1.4.1) and are distributed across the DBGlobe system. In the following section, we present how the application accesses the DBGlobe system and what kinds of services are provided by it.

4. User's Manual

This section is (i) a manual that guides users on how to use the DBGlobe system, and (ii) a manual that guides users on how to use the context aware application prototype SpotMe. In both cases a web based interface is provided to allow access by any kind of device; no installation is required.

4.1 DBGlobe System Manual

Users can have access to the DBGlobe system by creating a DBGlobe account. With this account, users can have access to web based interfaces that enable them to do the following:

1. Create an account
2. Login
3. View the available communities and the services that are registered to these communities.
4. Register its services.
5. Find any service that matches a set of keywords.
6. Use any application that has been created on top of the DBGlobe system.

A short manual for each of the above processes follows.

Creating a new account

Any user who wishes to have access to the DBGlobe system must own a DBGlobe account. Through this account, the user specifies some basic information about his identity. Figure 4.1.1 presents the form with this required information.

Figure 4.1.1: Register to DBGlobe

Login process

Any user who wishes to use the DBGlobe system can have access to it after his login process. As it shown in Figure 4.1.2, to access the system, the user must enter his DBGlobe login and password.

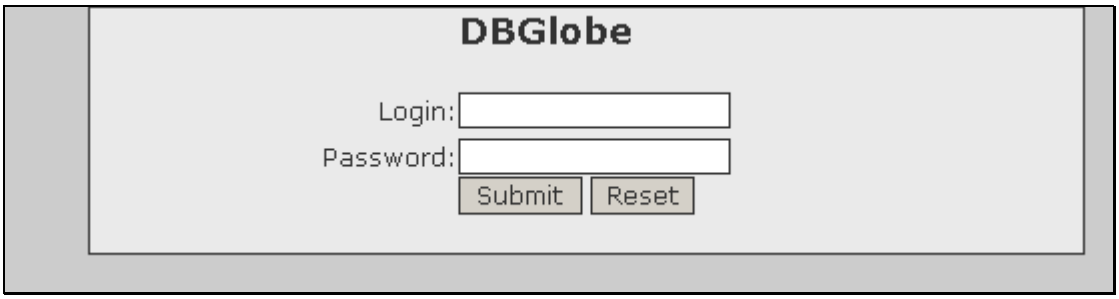

Figure 4.1.2: Login to the DBGlobe system

Using the DBGlobe system

After the login process the user is presented with the page shown in Figure 4.1.3.

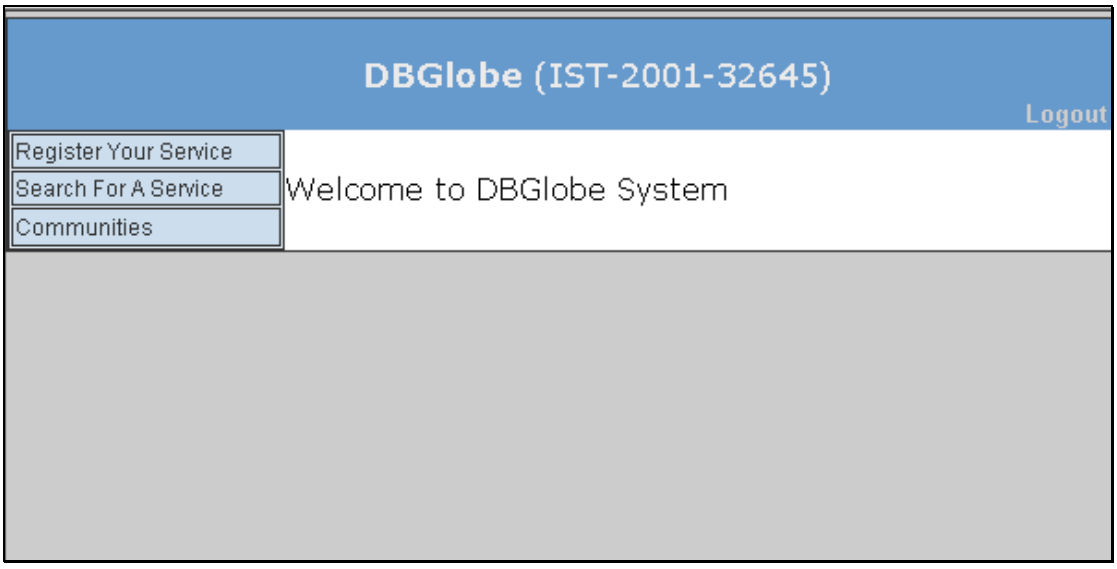

Figure 4.1.3: DBGlobe system interface

From this page, a user can click the links that are presented on the left of the page. Thus a user is able to:

- Register a new service
- Search for a service
- View available communities.

These actions are described next.

View Communities

Figure 4.1.4 presents all the available communities by giving their (i) uri (first column), (ii) description (second column) and (iii) registered to them services (fourth column).

i. The uri is necessary in case that the user wishes to have remote access to the community through his own application, and that is because of the fact that communities are distributed and can be accessed only by using their uri.

ii. Description, gives a notion of what kind of services are grouped under the umbrella of the specific community

iii. The set of services that are registered to the community are links to their wsdl descriptions. Users must be aware of the interface of the service that is presented through that description, in order to use the service. In case of an AXML service, the user will be able to call the service by clicking the link.

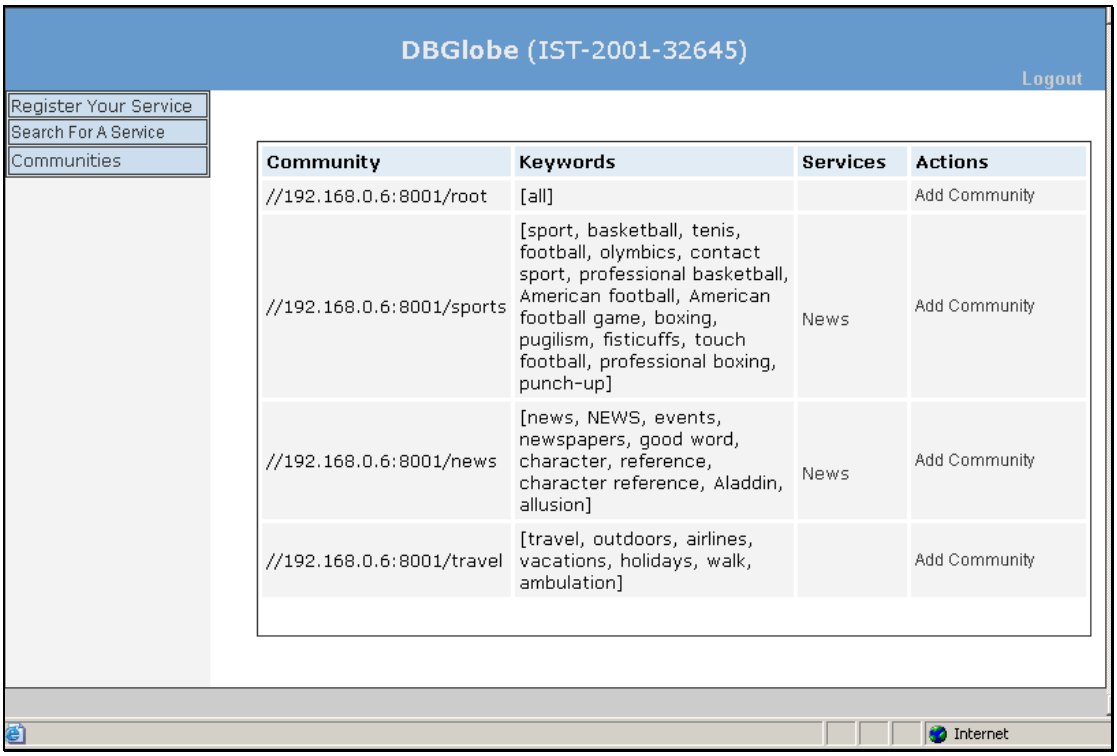

Figure 4.1.4: Communities

Register a service

The user is able to register his services to the system using the form of Figure 4.1.1. He must provide the name of the service, a set of keywords that describes the service, the running url and the WSDL url.

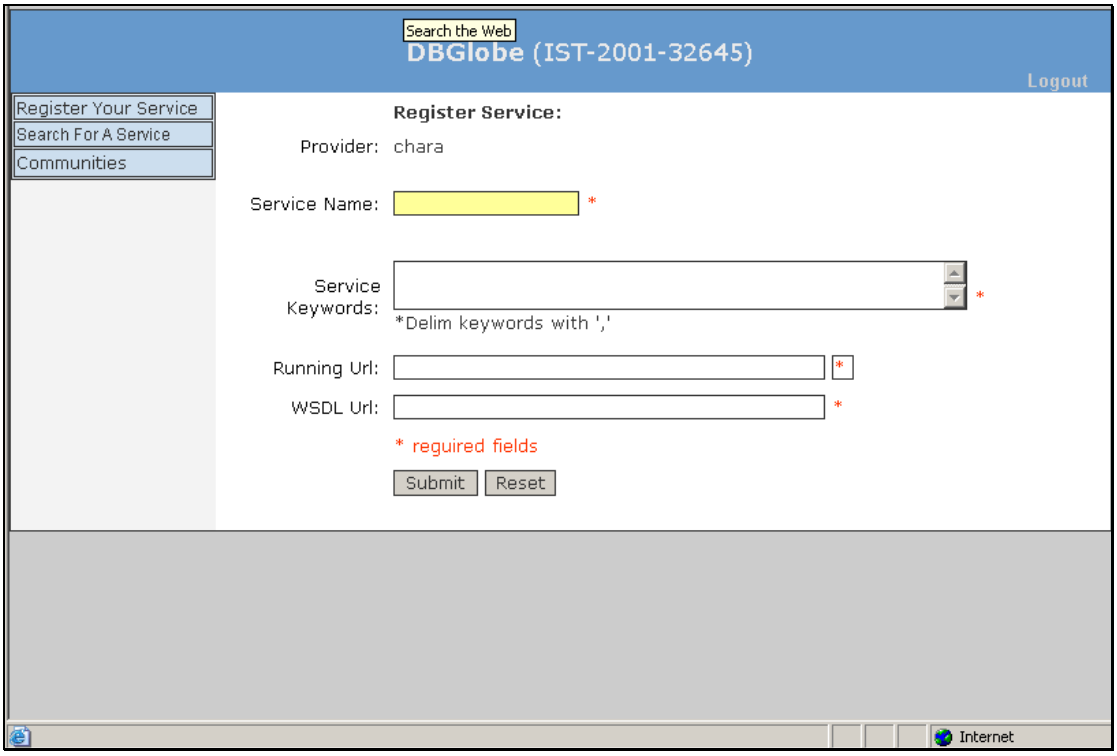

Figure 4.1.5: Register a service

Find a service

The DBGlobe system provides through the form of Figure 4.1.6 the capability to find a service. In order to find the service, the user must enter a set of keywords that describes the characteristics of the services that he wishes to find (e.g. if the user wishes to find the available weather services, he can fill the box with keywords like "news" and press the submit). The request will be forwarded to a CAS which will forward the request to the communities that match \the given keywords. The communities will execute the query ("*Select* services *where* description *contains* 'news'"). The keywords must be separated by using ",".

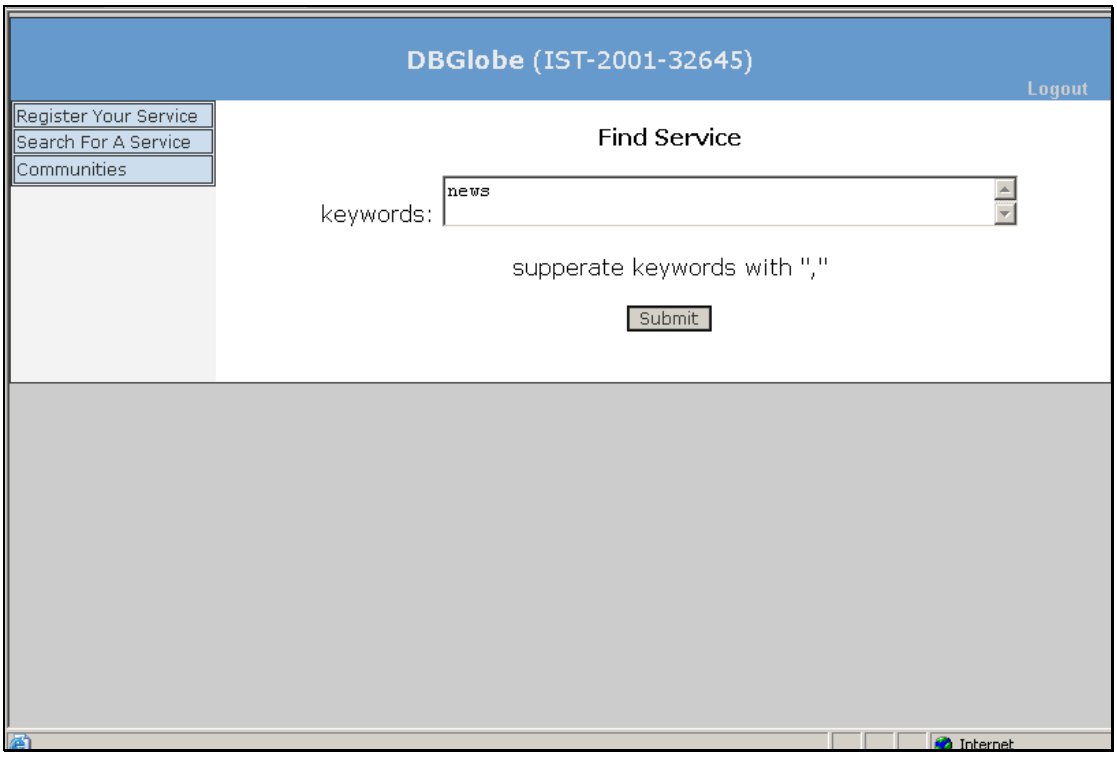

Figure 4.1.6: Find a Service

View service search results

In Figure 4.1.7, the user can see the results of the search request. The results contain the

(i) Service name. Users must be aware about the service name in order to invoke it through their applications.

(ii) Provider. Provider name is important in case we wish to provide quality of services. A user may prefer services that are provided by a specific user because the last is reliable.

(iii) Link to the wsdl of the service. As mentioned earlier the wsdl description is very important in case that user wish to invoke the service through his applications. In that case he has to be aware about the service profile in order to invoke the appropriate service.

(iv) Set of keywords. Those keywords give an outline of the service characteristics.

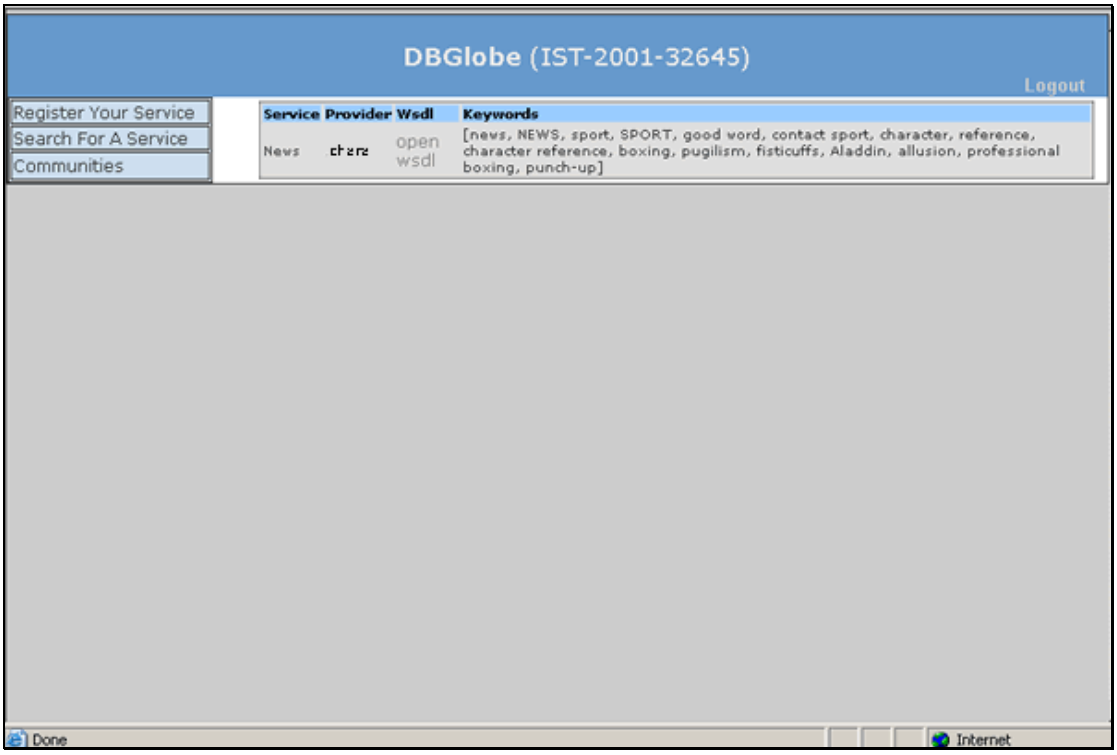

Figure 4.1.7: Search Results

Logout

At the top right of each page the user can select the logout link to exit from the system.

4.2 Manual for the Proof-of-Concept Prototype

In the previous paragraph, we presented a manual for using the DBGlobe system functions. In this paragraph, we present the manual for using the proof of concept prototype.

The major objective of the prototype is to create a tool that enables a user to exchange their services with the people they wish. The prototype gives the ability to

- Create a group of users
- To add/delete users
- To register user services
- To search for services
- To search for services among those users
- To find local services

In the following paragraphs, we present a short manual for the above.

Register to the system

Each user in order to use the prototype application must be a user of the DBGlobe system too. This is important because as we mentioned earlier the context aware application implements only the application specific functions (i.e. who is friend to whom), for the remaining functionality, (e.g. get user location, register service) it makes calls to the DBGlobe system. For example, in the case that the application is requested to find the location of a user, it must forward the request to the DBGlobe system, if the user is not member of the DBGlobe then the request would not be answered. Figure 4.2.1 presents the registration form.

Figure 4.2.1: Register to the system

Login to the application

Each user in order to login to the application must provide his login and password as shown on the Figure 4.2.2.

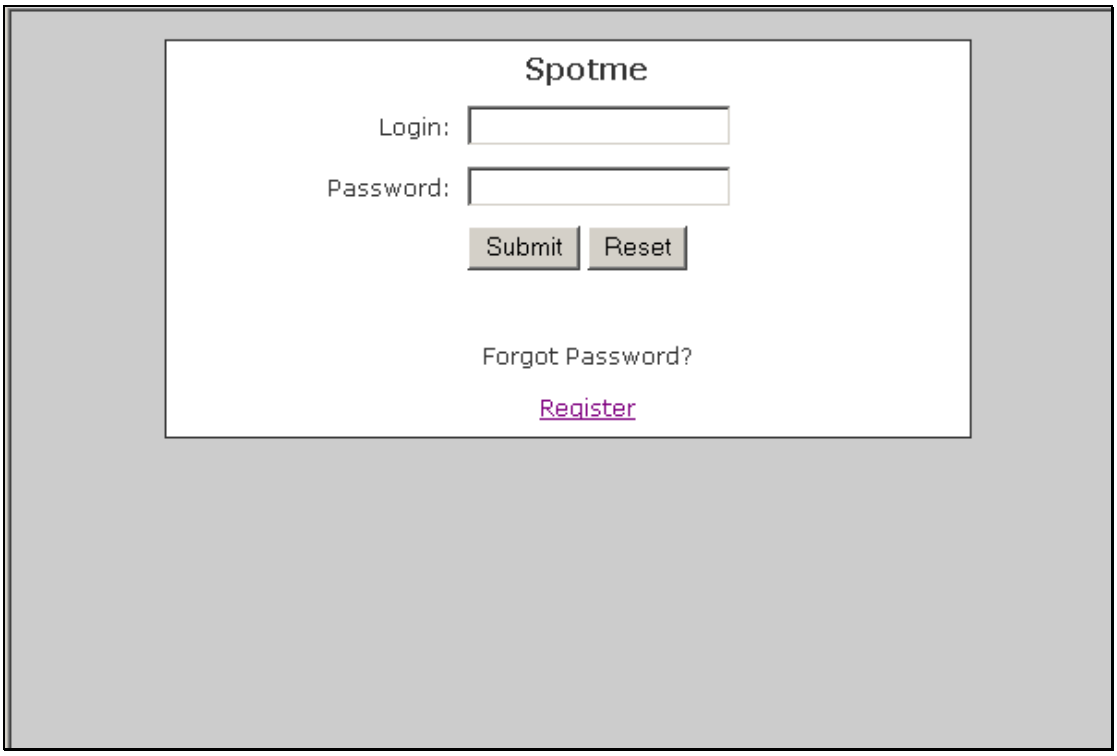

Figure 4.2.2: Login to the application

View your group and friends

Figure 4.2.3 shows the major screen of the application. The user can see its group of friends/colleagues/partners. For each member of the group, the information displayed includes:

- His status (online or offline), online users have an orange signal; offline users have a black signal.
- The name of friend.
- User location if available.

Each user name is a link to the user profile.

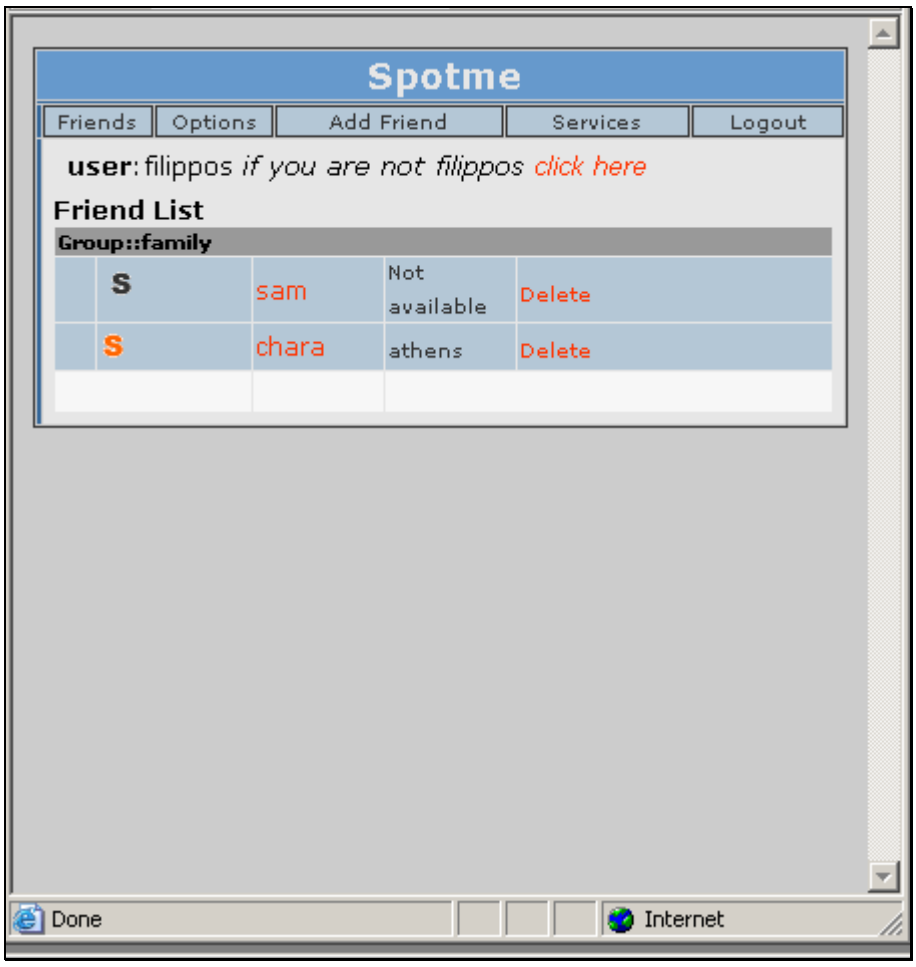

Figure 4.2.3: Major application screen

View friends/partner profile

Figure 4.2.4 exhibits users profile with the basic information. Additionally there is a link that allows user to view the services that the friend provides.

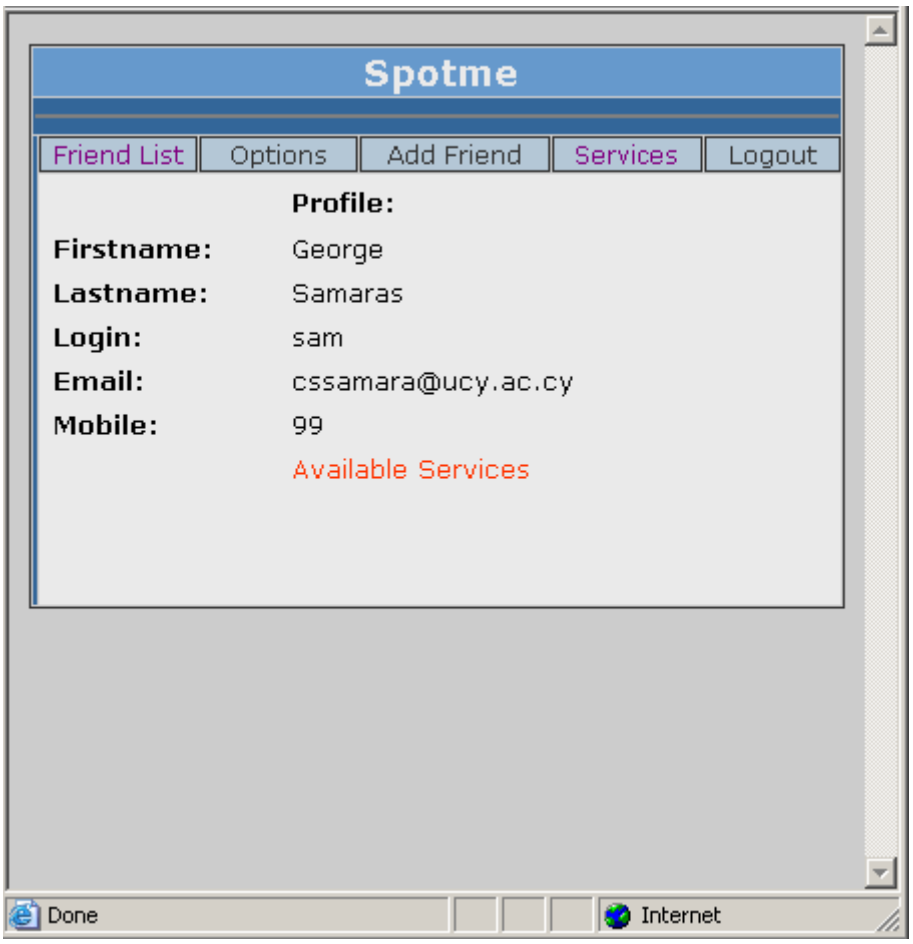

Figure 4.2.4: User's profile

Application Options

Figure 4.2.5 shows the two basics options of the application. Firstly, the user can be alerted when other users add him in their contact list. Secondly, the user can be alerted when a friend publish a new service.

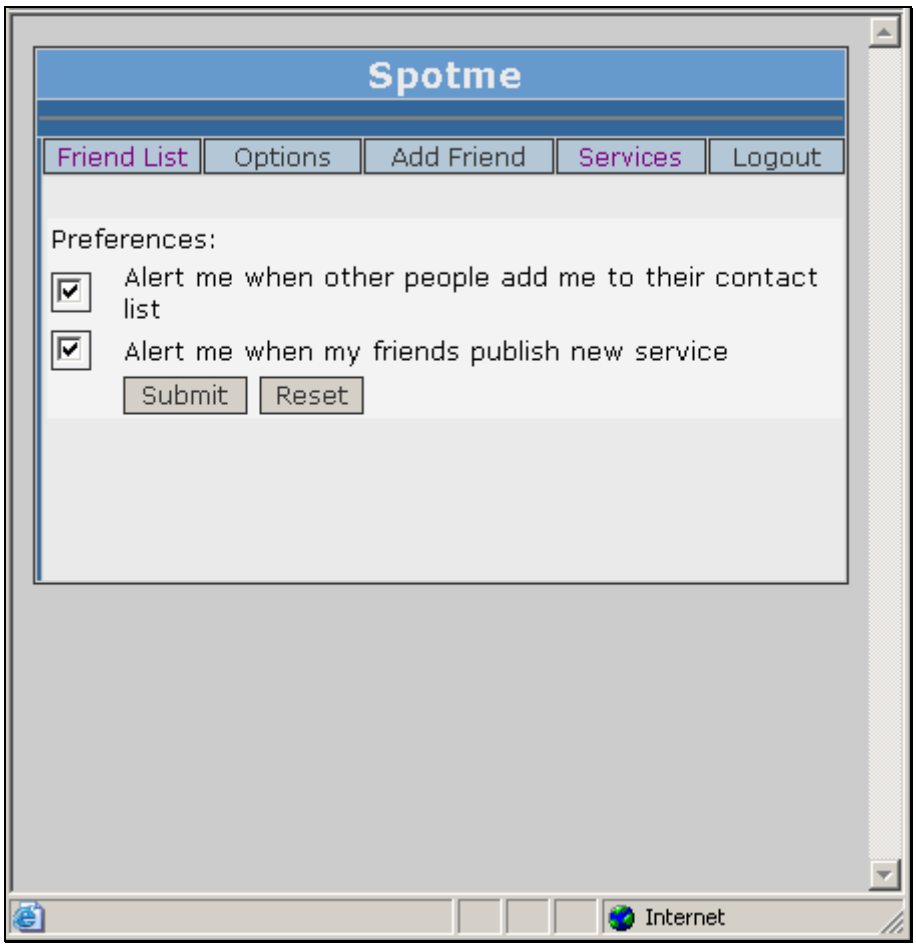

Figure 4.2.5: Application options

Friend Management

Figure 4.2.5.1 exhibits the page that opens by clicking the link "Add friend" at the top menu.

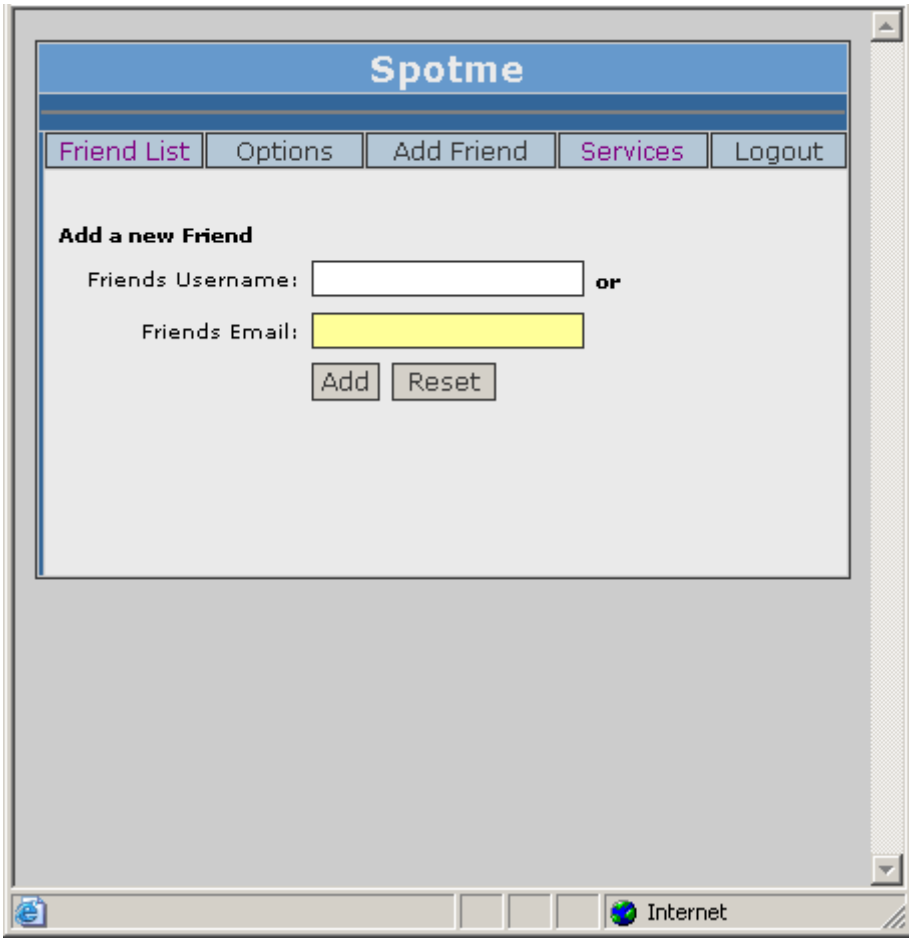

Figure 4.2.6.1: Add a new friend

Figure 4.2.6.1 shows the form that is available to add a new friend by giving either the login of the new friend or the email. The result of the form filling is the shown in Figure 4.2.6.2 where the user can select the user that wishes to add to his list by clicking the "select" link.

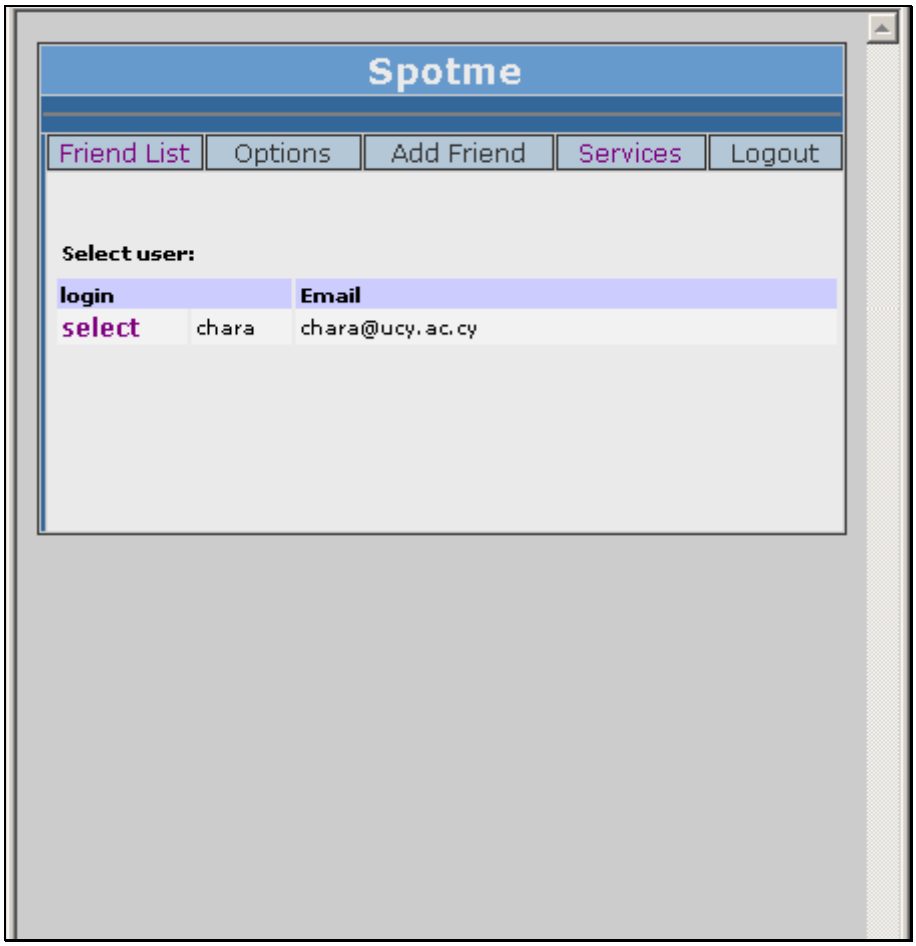

Figure 4.2.6.2: Select the new friend.

The next step is to select the group to which the new friend will be added. Finally the friend will be added to the list. If the new friend wishes to be informed when someone adds him to his list then the friend will be added only after the new friends allows it.

Service Management

At the top menu the user can select the link “services”. This link will open the page that is shown in Figure 4.2.7.

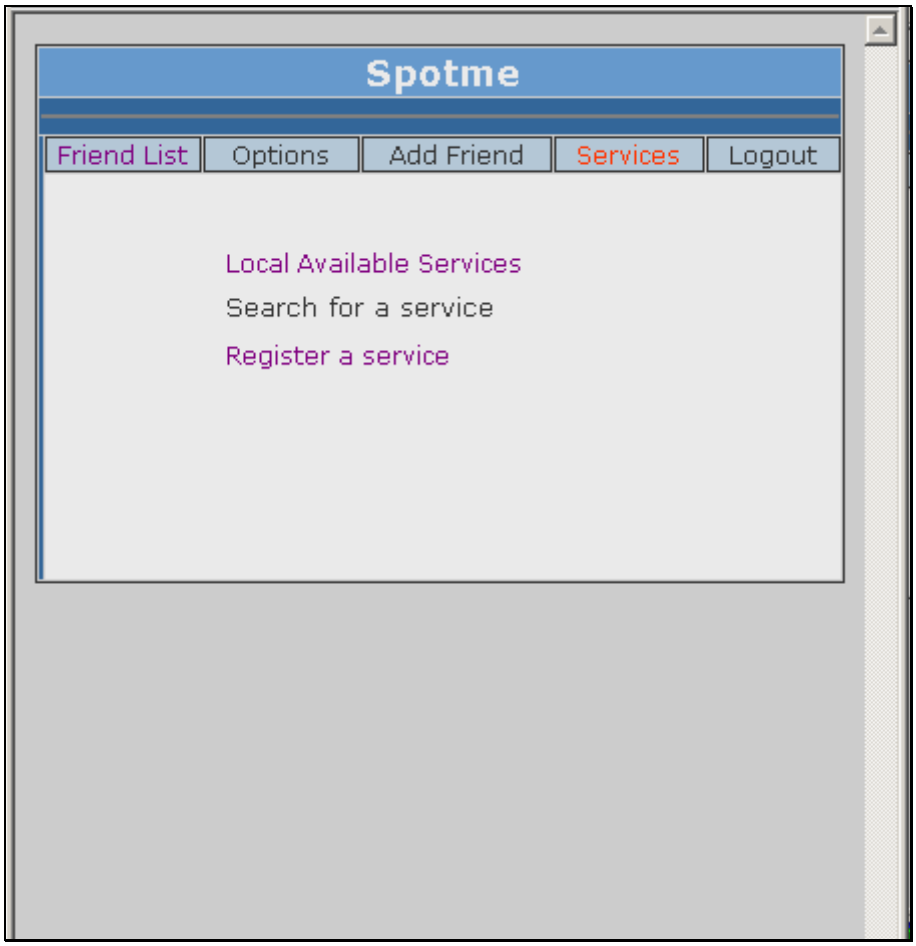

Figure 4.2.7: Manage Services

From this page, a user can follow either the link that shows the local available services, or the link that allows to user to search for a service and finally the link that enables users to register their service.

Search for services

The page shown in Figure 4.2.8.1 opens after the user has click to the link "Search for a service" of Figure 4.2.7.

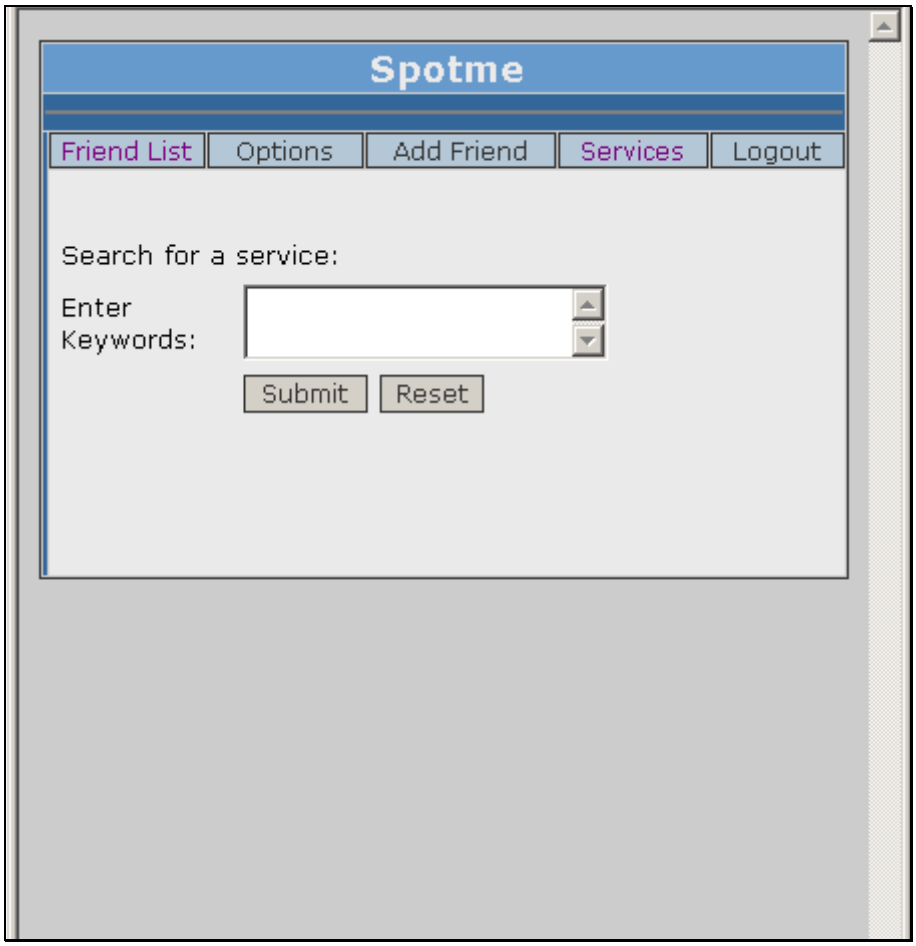

Figure 4.2.8.1

In the form of Figure 4.2.8.1, a user can enter a set of keywords that describe the service that he wishes to find. Figure 4.2.8.2 shows the results.

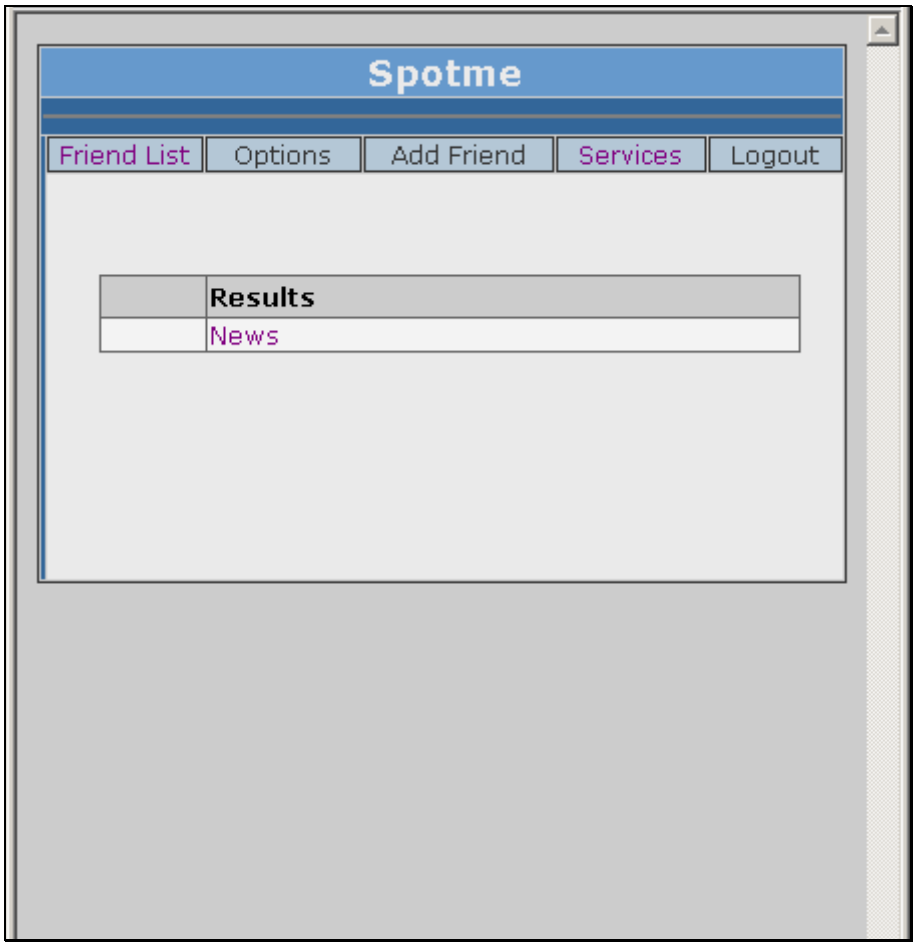

Figure 4.2.8.2: Search Results

By clicking to the service name the user can open the service in a new window.

Register a new service

The user can fill the input fields of the form (Figure 4.2.9) that opens when he clicks the link "Register a service" in Figure 4.2.7. He must provide (i) the Service name, (ii) the keywords that give an outline of the service characteristics, (iii) the end point url this is where the application runs, (iv) the wsdl url (this url will be used by the user in order to be aware about the service interface).

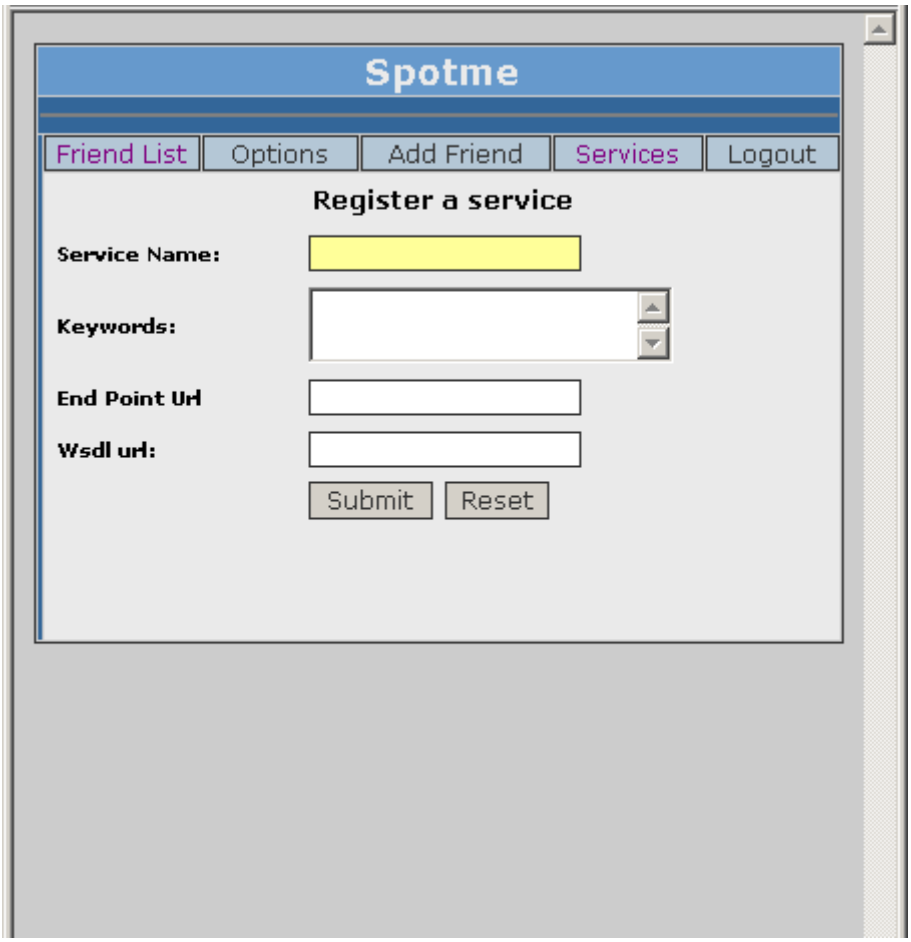

Figure 4.2.9: Register a new service

Logout

Finally, the user can click the link “logout” of the top menu to exist the system.

Conclusions

In this deliverable, we describe the proof of concept prototype. In detail, we showed how communities can help answer not only location based queries but semantic level ones in an organized and efficient manner. The fact that communities define/create ad hoc databases around "concepts" and not just around location enabled our prototype application to achieve itss goals. That is, communities enabled SpotMe to find and keep track of all "friends" that have similar interests with a given user. Finally, as shown, in order to enable keeping constant track of friends we needed to utilize continues queries. It is clear that without the utilization of communities some tasks would be extremely difficult to implement within the scope of a relatively small application.

To summarise, in this deliverable we presented SpotMe, an application that utilizes DBGlobe communities in order to create and support real user communities.

Bibliography

[1] G. Koloniari and E. Pitoura, "Bloom-Filters for Hierarchical Data", Proceeding of the 5th Workshop on Distributed Data and Structures (WDAS), 2003.

[2] Services Definition Language (WSDL), *Web page*, http://www.w3.org/TR/WSDL.

[3] Ouzzani, M., Benatallah, B., and Bouguettaya, A.: Ontological Approach for Information Discovery in Internet Databases. *Distributed and Parallel Databases Journal*, 8:367-392, 2000.

[4] S. Abiteboul, O. Benjelloum and all. Active XML: Peer to Peer data and web service integration.

[5] Simple Object Access Protocol (SOAP), Web page, http://www.w3.org/TR/SOAP/

[6] Deliverable D3: Overall system architecture

[7] Deliverable D7: Final Simulator prototype and Simulator Web Page

[8] Deliverable D11: The query language of DBGlobe

[9] Deliverable D12: Query optimization and execution in DBGlobe.

[10] Deliverable D14: Location Aware Ad-hoc Databases and Query Processing and Beyond.

[11] http://ws.apache.org/axis/